個々のコンピュータ特性を考慮したプリプロセッサ用の最適化コードジェネレータ(数理計算技術の基礎理論)

AUTHOR(S):

内田, 智史; 八巻, 直一; 本郷, 茂; 唐澤, 豊

CITATION:

内田, 智史 ...[et al]. 個々のコンピュータ特性を考慮したプリプロセッサ用の最適化コードジェネレータ(数理計算技術の基礎理論). 数理解析研究所講究録 1993, 832: 151-161

ISSUE DATE:

1993-04

URL:

http://hdl.handle.net/2433/83383

数理解析研究所講究録
第832巻 1993年 151-161

個々のコンピュータ特性を考慮したプリプロセッサ用の最適化コードジェネレータ

神奈川大学 内田智史 (Satoshi Uchida)
システム計画研究所 八巻直一 (Naokazu Yamaki)
専修大学 本郷 茂 (Shigeru Hongo)
神奈川大学 唐澤 豊 (Yutaka Karasawa)

1. はじめに

プリプロセッサやコードジェネレータは、少ない労力で比較的高度な機能を実現できるので、特定問題向き言語や簡易言語などを実現する手段としてよく用いられている。我々も、この利点を活かしてLAMAXと呼ばれる一連の行列演算用言語を開発してきた。しかし、一般に、プリプロセッサは、スピードを犠牲にして機能性を追及したシステムであると考えられており、これまでプリプロセッサの高速性についてはあまり議論されてこなかった。とくにプリプロセッサの最適化については、ほとんど議論されていない。これは、プリプロセッサが、通常FORTRANやCなどの高級言語をそのオブジェクトとして出力するため、機種依存性を極力低く抑えることができても、その反面、ハードウェア命令をフルに使うという最適化ができないことに起因している。

我々は、1989年に開発を開始したLAMAX-S[1]のコード生成において、プリプロセッサレベルの最適化を試みている。LAMAX-Sの言語仕様は、数学的な式の変形や行列の数学的性質から、プログラム変換を施すことによりより計算量の少ない手順で実行できるような最適化が可能なように設計された[応用数理学会文献参照]。現在、我々は、この処理方式を実用化した最適化処理系を開発中である。

この最適化を我々は数学的最適化と名付けたが、LAMAX-Sでは、この他に、各コンピュータの特性に合わせたチューニング作業の自動化を計画している。

一般に、アルゴリズムをそのままストレートに表現したプログラムよりも、高速化のためにハードウェア／ソフトウェアの特性を利用してコーディングしたプログラムの方が速くなることが以前から知られており、数々のコーディング手法が考案されている。

近年、スーパコンピュータの普及と共に、チューニングと呼ばれる作業が日常化するにつれて、このような認識が特に強まっている。たとえば、チューニング手法の中で比較的よく紹介されているものにアンローリングがある。これは、多重ループに用いられる外側のループの制御変数のステップ幅を2以上にしてループの回数を減らす方法である。これは、ベクトル計算機以外でも効果があり、たとえば、スーパコンピュータが登場する以前に作成されたLINPACKなどでも行われている。実数型の行列の乗算にこれを適用した例では、SUN(Sparc)では80%、パソコン(PC9801RA)では70%程度の実行時間の短縮効果がある。スーパコンピュータの場合、アンローリングの効果は最大で50%程度と非常に有効である。

また、最近のワークステーションのCPUは、RISCで構成されているものが多い。このタイプのコンピュータでは、主記憶、キャッシュ、レジスタ、演算器間をデータ移動させる時間が演算時間そのものよりも長くなる傾向があり、既存のプログラムの作成方法では必ずしも高速なプログラムを作成できない可能性のあることが指摘されている[2]。

つまり、プログラムの高速化を追及するには、CPUの特性も重要な要因になっている。

プリプロセッサの生成するオブジェクトコード(FORTRANやCなどのプログラムコード)の高速化を考えたときに、このようなチューニング処理は避けて通れない問題である。

プログラムの実行効率を高めるコーディング方式は、本来、メモリのインタリーブ数やキャッシュのサイズ、演算レジスタや演算器の個数、演算パイプラインの方式などによって異なる。このようなコーディング方式は、ハードウェアの仕様からある程度推量できる。しかし、かなり厳密にハードウェアをモデル化しないと正確な推定は困難であるし、また、予想外の落し穴も多い。その理由の一つとして、プログラ

ムの実行速度が、コンパイラの特性にも影響されるという点が挙げられる。特に最適化コンパイラの場合、このことが顕著となる。したがって、高水準言語を生成するプリプロセッサの場合、その次のフェーズとなるコンパイラの特性を知ることが重要である。

本論文の目的は、LAMAX-Sが生成する行列演算プログラムのコード生成において、実在するハードウェア／ソフトウェア上で実際に高速となるチューニング法を与える実証主義的かつ現実的な方法を提案し、その実例を示すことである。また、本方式で対象とするコンピュータは、スーパコンピュータ、汎用コンピュータ、ワークステーション、パソコンである。

具体的には、本方式では、次の手順で各コンピュータのアーキテクチャに合わせたチューニング処理方式を決定する。

(1) コンピュータのアーキテクチャに合わせて計画された計測テストプログラムとそれを起動するためのJCLあるいはコマンドを自動生成する。(2) 計測テストプログラムを実行し、データを得る。(3) その実施結果の分析結果から、測定対象コンピュータに対するチューニング方法を表現したルールベースを作成する。

LAMAX-Sプリプロセッサは、このルールベースに従って、各コンピュータ用にチューニングしたオブジェクトコードを生成する。

2. チューニング項目とテストプログラム生成

現在では、さまざまアーキテクチャに基づくコンピュータが実用化され実際の計算に用いられている。たとえば、多くのスーパコンピュータで用いられているパイプライン方式のコンピュータ、IAP(内蔵型アレイプロセッサ)をもつ汎用コンピュータ、SUN SPARCチップのようなRISCコンピュータ、トランスピュータのような並列コンピュータ、あるいは、計算を実行するタスクをソフトウェアで並列化し複数のCPUで実行するCONVEXのようなコンピュータ、数値演算プロセッサを付加したインテル8086系のパソコンなどがある。

これらのコンピュータのハードウェアの特性を調査すること、さらにその各々のコンピュータ上に登載されているコンパイラの個々の特性を調査することは膨大な作業量となる。そこで、今回提案する方法では、まず、対象となるチューニング手法を絞り込み、列挙することから始める。本論文では、行列演算をその題材として取り上げているので、(1) 乗算(アンローリング・do文の形式)、(2) do文の多重化、(4) 行列のベクトル化、3項目を調査対象とし、これらのテストプログラムを生成するシステムを作成して実験した。

調査対象項目について簡単に述べる。

乗算では、アンローリングおよびdo文の形式について調査する。この手法は、前述のようにベクトル計算機の場合には、複数のベクトルを同時に処理することが可能となり、かなり効果があるが、それ以外の計算機でもメモリとのデータ転送の軽減やレジスタの有効利用などの利点がある。一般には、アンローリングの分割数は2〜4が最適であると言われているが、これはコンピュータの種類によってもまた、対象となるデータ型やデータ量によっても微妙に異なる。各型に対してさまざまな分割数についてのテストプログラムを生成し、分割数と実行効率の関係を調べる。なお、アンローリングでは分割数が多くなると、プログラムコードが増えるのが欠点である。do文の形式とは、乗算を処理するdo文の制御変数の配置順序である。たとえば、次の左の例をKJI方式と呼び、右の例をJKI方式と呼ぶ。

```
   do 10 k=1, 100
     do 10 j=1, 100
       do 10 i=1, 100
         A(i,j)=A(i,j)+B(i,k)*C(k,j)
10 continue
```

```
   do 10 j=1, 100
     do 10 k=1, 100
       do 10 i=1, 100
         A(i,j)=A(i,j)+B(i,k)*C(k,j)
10 continue
```

do文の多重化とは、1つのdo文中に文を極力まとめて記述した方がよいのか、それとも別々に1つの

do 文中には 1 つの文のみを記述した方がよいのかを調査するものである。

行列のベクトル化とは、2 次元配列である行列を、EQUIVALENCE 文によって 1 次元配列化して、ベクトル長を増やす手法である。

これらの手法の効果を測定するテストプログラムを作成して実行し、その結果を分析すれば、その手法に対する効果が得られる。しかし、先にチューニング項目を絞り込んだとはいえ、これらの項目をすべての要因に渡って調査することは大変な作業量となるし、またその実行に要するコストも膨大なものとなる。そこで、あらかじめ測定したい項目を選択しその指示に従って、テストプログラムと対象オペレーティングシステムの JCL あるいはコマンドが生成されるプログラムを作成した。

次の例は、スーパコンピュータ S820 におけるテスト項目指示書 (一部) である。この指示書に従って、計測プログラムが作成される。

```
Machine:S820
OS:VOS 3
Compiler:FORTRAN/HAP
Option:SOURCE,SOPT,HAP(DIAG(FULL),ANALYZE(SOURCE))
declaration:REAL*8 scalar, vector
start:CALL XCLOCK
start:CALL VCLOCK
terminate:CALL XCLOCK(scalar)
terminate:CALL VCLOCK(vector)
calculation:write(*,'('' '',F18.8,'' '',F18.8)') scalar,vector
test:item=(mult,unrolling),type=r8,size=50,dotype=KJI
test:item=(mult,unrolling),type=r8,size=50,dotype=JKI
test:item=(mult,unrolling),type=r8,size=100,dotype=KJI
                    :    省略
```

この他に、これらの各プログラムを実行するための JCL を生成するために、次のファイルを作成する。この中で##が各テスト項目を識別する識別子となる。FORTRAN における時間計測などに関するファイルがある。このテストプログラムならびにテスト用の JCL を作成するプログラムを作成する作業は、非常に単純な作業であり、他の種類のチューニング項目を調べることもそれほど困難なことではない。

```
MEMBER:@BATCH.CNTL:COM##
JCL(COMPILE)://G.....## JOB ********,CLASS=A
JCL(COMPILE):>>USE @TEST.FORT,@TEST.LOAD
JCL(COMPILE):>>COM PGM##,PARM($$)
JCL(COMPILE)://
MEMBER:@BATCH.CNTL:GO##
JCL(GO)://G.....## JOB ********,CLASS=B
JCL(GO)://*MAIN SYSTEM=SUPER
JCL(GO):>>USE ,@TEST.LOAD
JCL(GO):>>FILE FT06F001,@TEST.DATA(OUT##),ABS,SHR
JCL(GO):>>GO PGM##
JCL(GO)://
```

また、ここに挙げたチューニング手法以外にも、メモリインタリーブなどの影響を考慮することが重要であるが、これは、比較的、ハードウェアのバンク方式の構造から推定できるので、あえてテストプログラムからははずした。

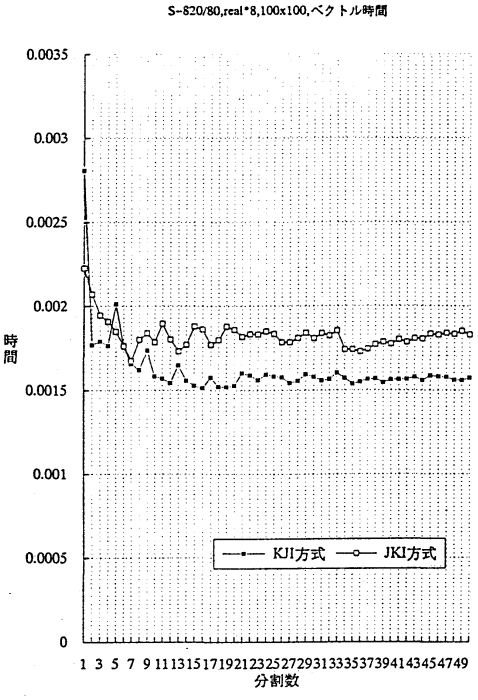
3. テスト結果例と分析例

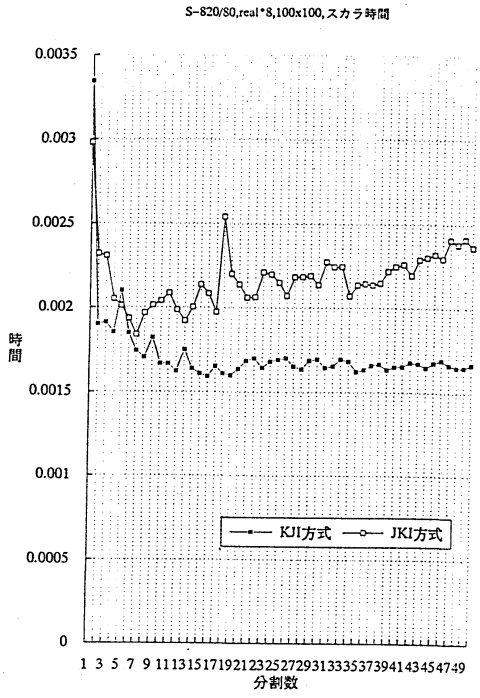
本方式の適用を試みたのは、次のコンピュータである。スーパコンピュータとしては、日立の S820(東大大型計算機センタ)、日電の SX-1EA(青山学院大学)、富士通の VP30(神奈川大学) である。汎用コンピュー

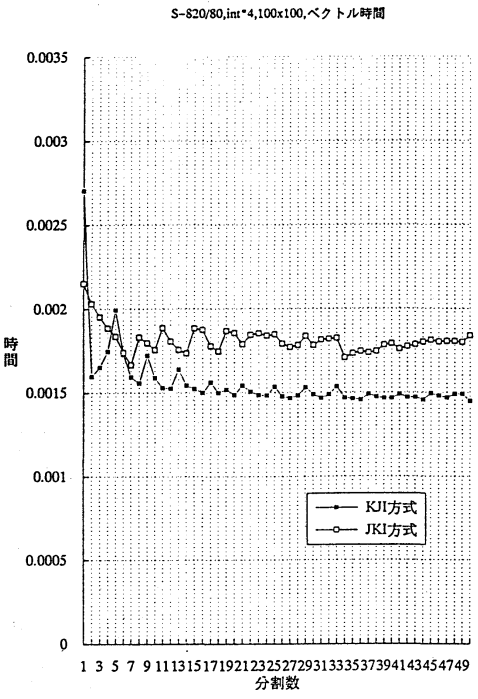
タとしては、日立 M880H(東大大型計算機センタ) と富士通 M770(神奈川大学) である。ワークステーションとしては、CONVEX である。パソコンとしては、PC9801 RA である。

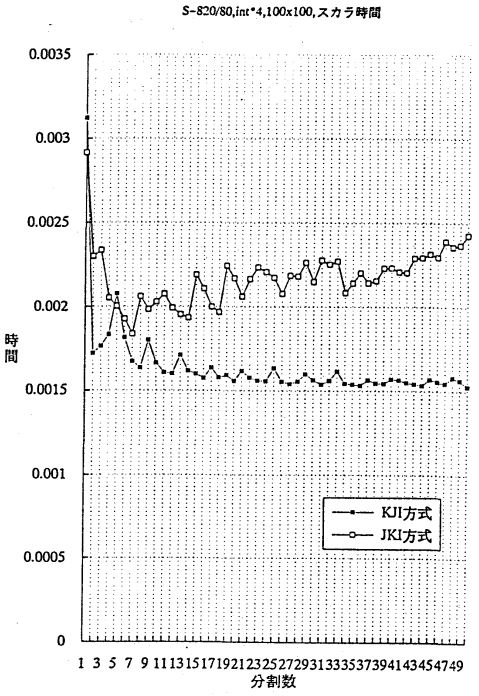
3.1 乗算：アンローリング・do 文の形式

乗算におけるアンローリングは、ほとんどの機種で効果があるが、唯一、CONVEX は例外であった。スーパコンピュータの場合、すべてにおいて非常に効果的である。次の図は、S820 における倍精度実数型および整数型の 100×100 の行列の乗算について KJI 型と JKI 型について調べた場合の表である。



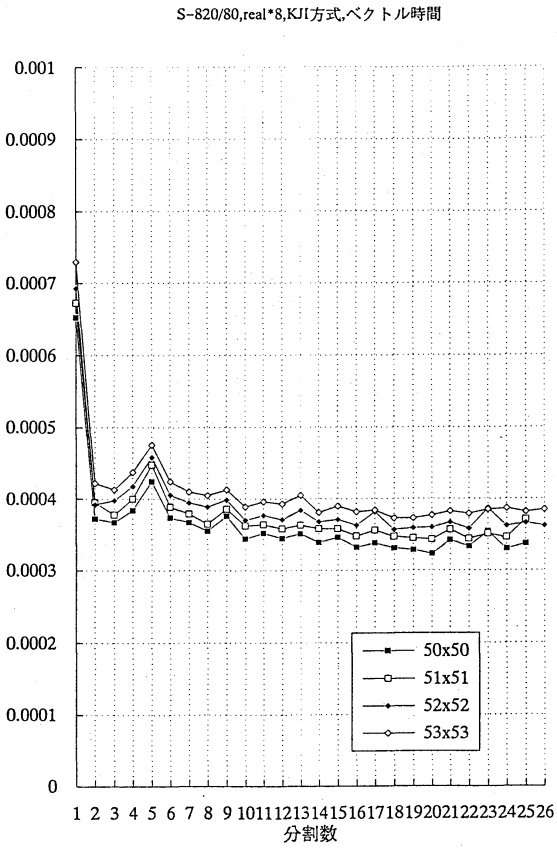


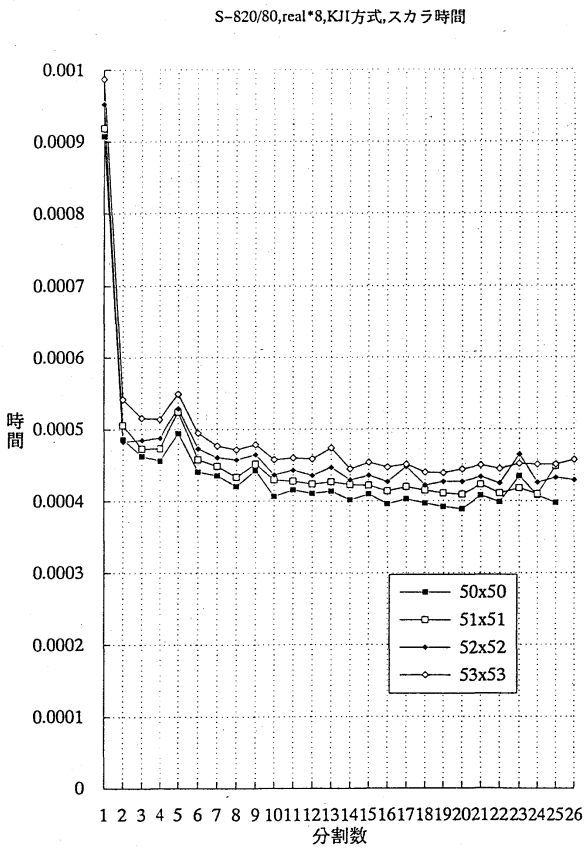




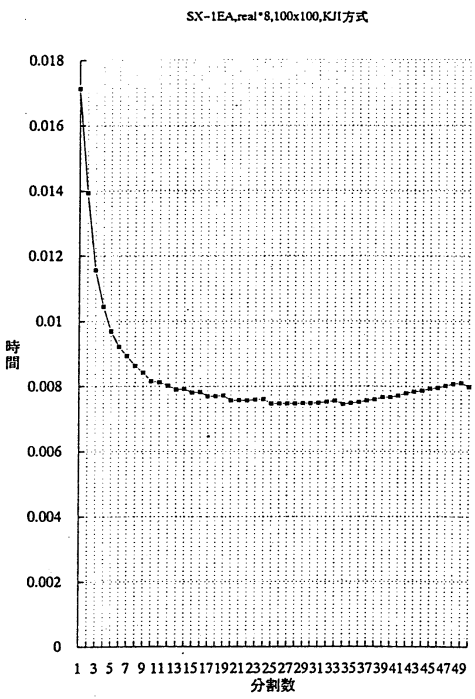
この場合には、倍精度実数型および整数型ともにKJI型の方が全体的に効率がよい。他の型のさまざまな寸法についての調査結果もKJIの方が効率がよいことを示している。このグラフでは、どちらの型についてもすでに分割数が2の場合にかなりの効果を示しているが、最も効率がよいのは16のときである。しかし、あまりに分割数を多くするとプログラムが大きくなり、メモリを消費してしまうので、記憶域を節約したい場合には、「分割数2あるいは8が適当」ということになる。

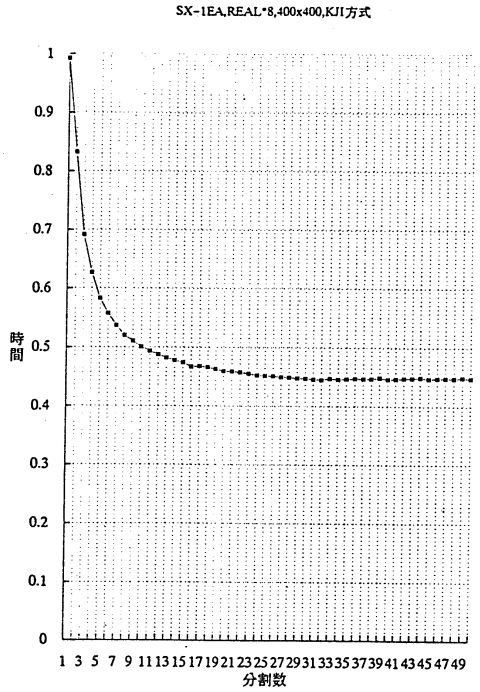
寸法の違いによる影響を見るために、寸法が50～53の場合のグラフを示す．これはKJI型の場合であるが、どれもだいたい同じ傾向を示している．寸法が52の場合だけが他と多少異なる結果となっている．このようにアンローリングの数値は、その寸法によってわずかであるが左右される。



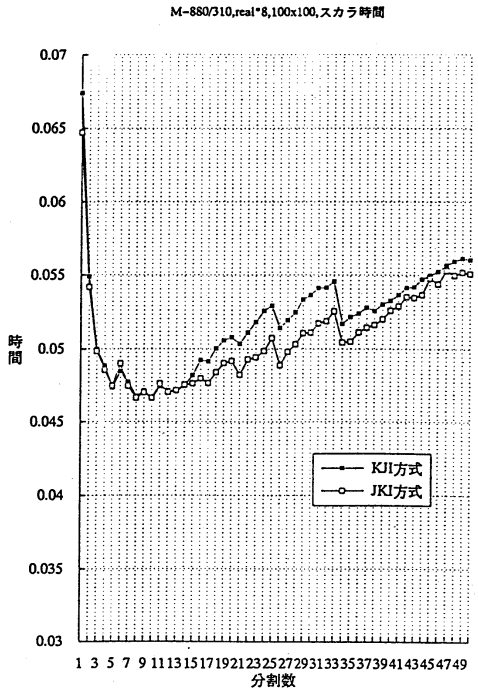


SX-1EA の場合にはさらに異なった結果となっている．以下の図は、スカラ時間とベクトル時間のトータルのグラフであるが、S-820 に比べてかなりなめらかなカーブを描いており、効率の良い分割数がかなり大きな値になっている．

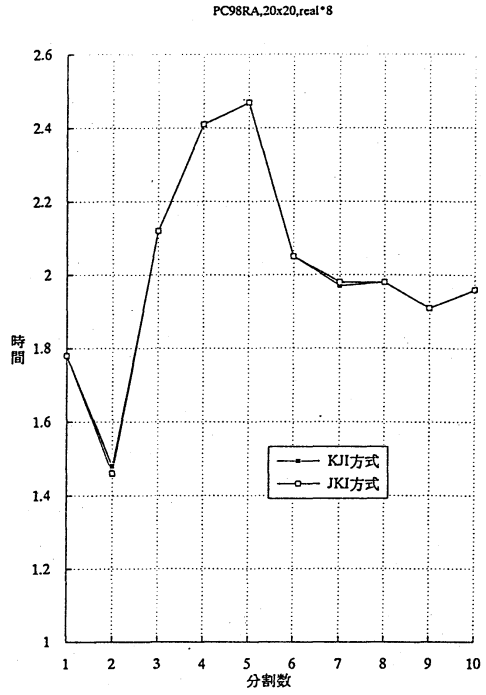


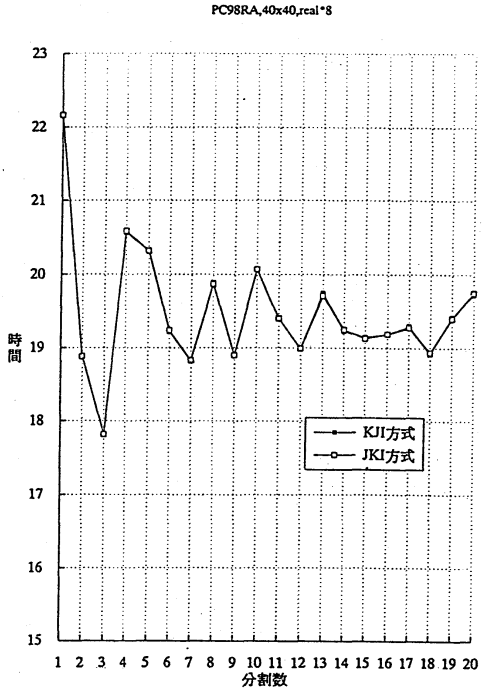


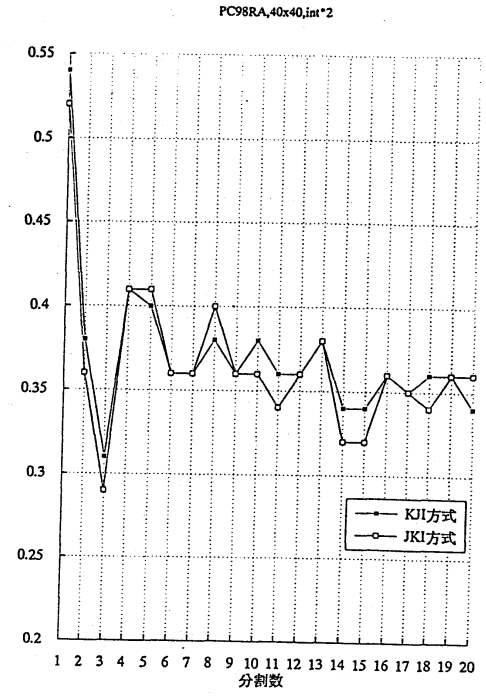
これが、汎用機になると、事情が異なってくる。M880 の例で考えてみよう。KJI 型と JKI 型では、わずかながらに JKI 方式の方が効率がよい．さらに、最初のピークが分割数 5 であり、最も効率の良いものは 10 である．



ワークステーションやパソコンの場合には、2〜3 が最も効果的である。以下にパソコン (PC98RA,MS-FORTRAN, 浮動小数点コプロセッサなし) の例を載せる．KJI 方式、JKI 方式の差はほとんどなく、寸法が 20 のときは分割数が 2 が最もよく、寸法が 40 の場合には分割数が 3 のときが最もよくなる．なお、整数型 (2 バイト) の場合にも同じ結果が得られた.

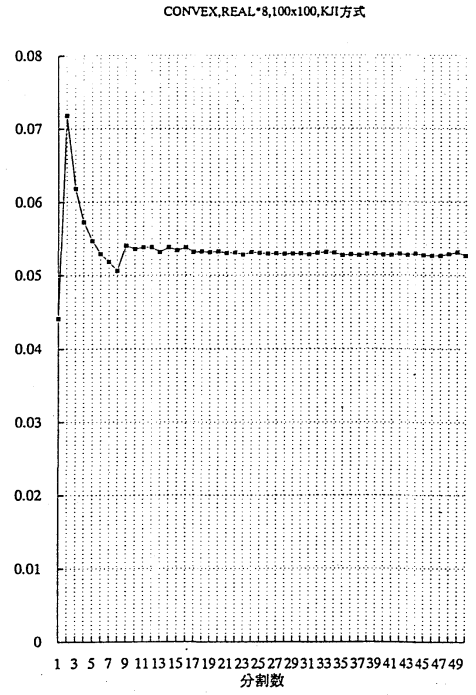


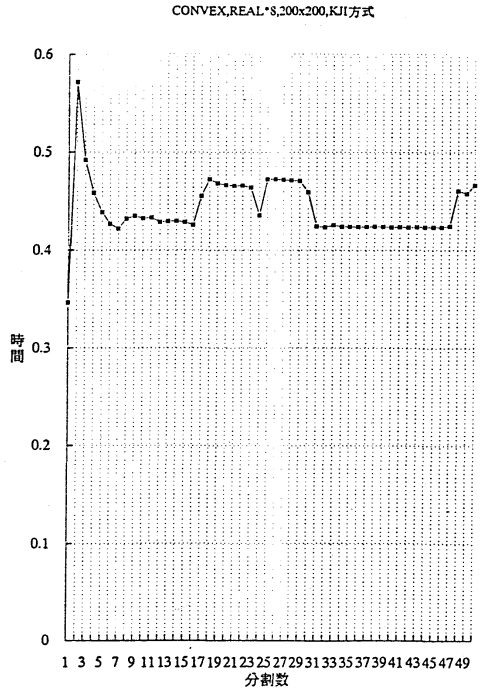




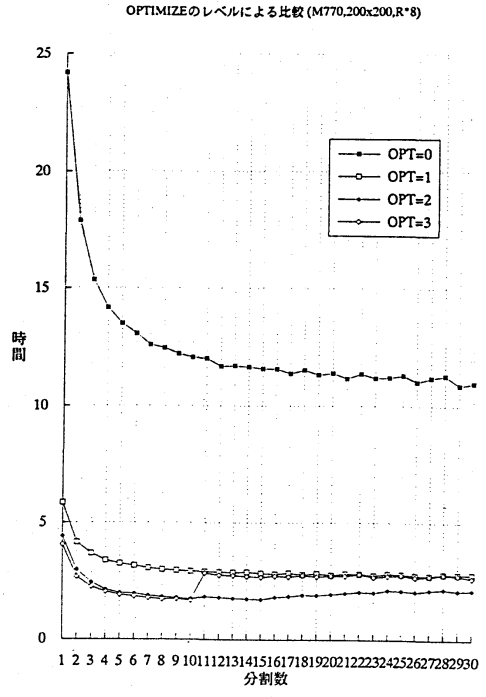
この調査の中で唯一の例外はCONVEXである。図にCONVEXのアンローリングの結果を示す。

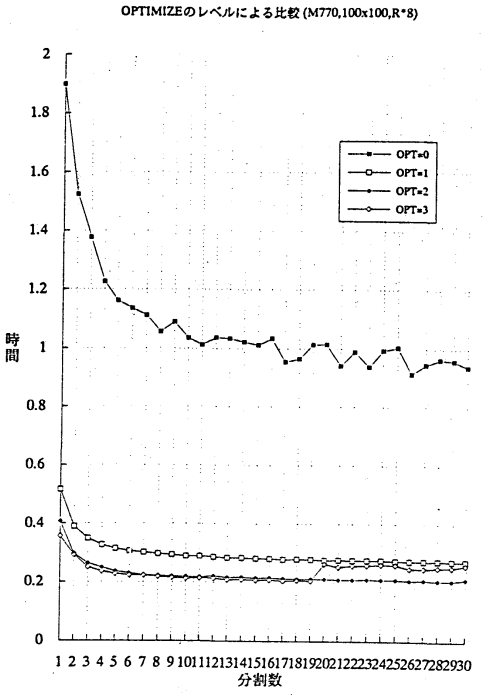
これはアーキテクチャが他のコンピュータと大幅に異なっているからであろう。CONVEXの場合には、アンローリングをしてはならないということになる。





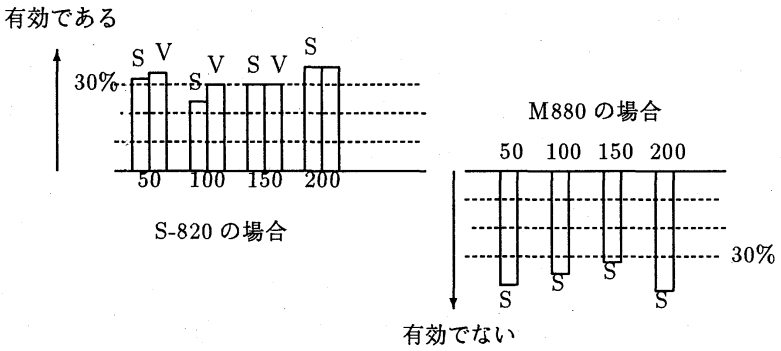
参考のために、コンパイラオプションの最適化の度合いによる例も調べてみた。





3.2 do 文の多重化

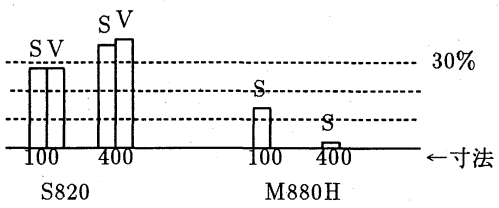
一般に、do 文の中に代入式を多く入れるといっても、さまざまなケースが考えられるので、本調査では、その傾向を捉えることしかできない。この調査では、400×400 の行列の加算を行う 4 つの式を 1 つの do 文にまとめたものと 1 つ 1 つ別の do 文にまとめたものを比較し、その結果を比で示した。スーパコンピュータの場合には、ほとんどの機種で do 文の中に多くの式を入れた方が効率がよくなる。しかし、汎用機である M880 の場合には、遅くなるという傾向が見られた。



しかし、それ以外の機種 (WS・パソコン) では、逆にわずかながら遅くなるケースが多い。

3.3 行列のベクトル化

2 次元配列で表現されている行列を高速化のために、1 次元配列で表現することは以前から行われていた。これは、2 次元配列よりも 1 次元配列の方が添字計算に要する時間が少なくて済むからである。スーパコンピュータの場合には、ベクトル長を長くするという利点がある。ところが、最近のスーパコンピュータのコンパイラは、この処理を自動的に行ってくれるので、あえてプログラム上で行う必要はないという認識が高い。しかし、我々の調査では、たとえば、S820 では、たしかにコンパイラが 2 次元配列を自動的に 1 次元配列化してくれる。しかし、プログラムが EQUIVALENCE 文を用いて 1 次元配列化した場合には、自動的に分割数 2 のアンローリングも行ってくれるので、それだけ速くなるという結果が得られた。スーパコンピュータ (S820) では 1 次元配列化していない場合に比べて 30%前後高速になる。汎用機の場合、スーパコンピュータほどではないが、わずかながら (数～10 数%) 効果がある。したがって、S820 の場合には、1 次元配列化は効果があり、さらに、そのループに対してアンローリングを施すとよいことがわかる。



しかし、パソコンの場合にはまったく差は認められなかった.

4. テスト結果例から導かれるチューニングルール

実際のルールを決める際にはたぶんに戦略的な考えが必要となる。たとえば、アンローリングを取り上げてみよう。次表に、S820 の 100×100 の行列の乗算に対する分割数 1 に対する比を示す。

表 100×100 の行列の乗算の計算時間の比 (S820)

分割数	1	2	3	4	5	6	7	8	9	16
比 (%)	100.0	59.7	60.5	58.9	67.0	58.1	54.8	54.0	57.3	50.8

分割数 2 と最高速を与える分割数 16 ではわずかに 8.9%しか違わない。分割数 16 の場合、かなりメモリを消費する。わずか 8.9%の高速化のために、メモリを犠牲にして良いのかどうかの判断は難しい。このような場合、分割数を 2, 8, 16 のどれかに取ることが考えられる。そこで、つぎのようなルールが生成される。(1) メモリを優先する場合には、分割数 2。(2) 速度を優先する場合には、分割数 16、(3) 両方を中立的に活かしたい場合には、分割数 8。

実際には、この他に do ループの型、要素の型、などもルールの中に指定される。

ルールの一般形は次の形をしている。

手法名 (検索条件：値, …, 設定条件＝値, …)

```
unrolling(size:100,type:r8,dotype=KJI,unroll(memory)=2,unroll(both)=8,unroll(speed)=16)
```

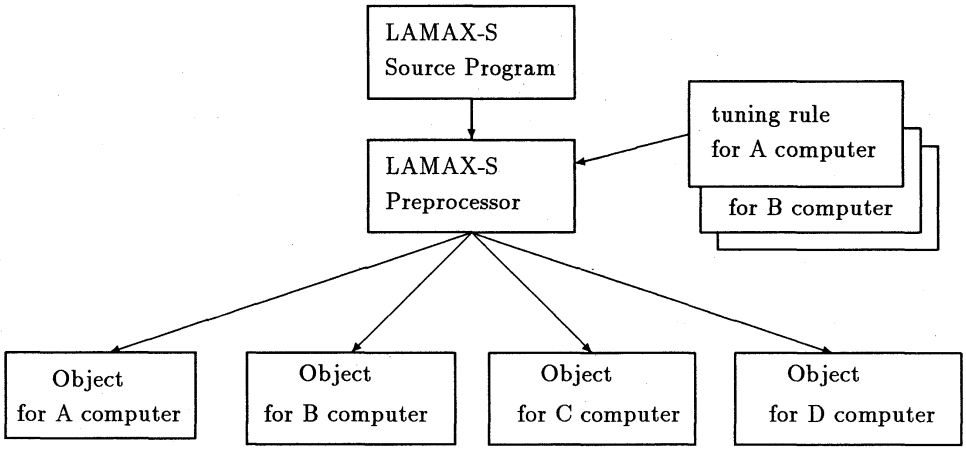
しかし、これではテストしていない寸法に対するルールが与えられないので、各要素の型ごとに既定のルールを設定する。

```
unrolling(size=(-50),type=r8,dotype=KJI,unroll=3)
unrolling(size=(51-),type=r8,dotype=KJI,unroll=2)
unrolling(size=*,type=i4,dotype=KJI,unroll=2)
```

5. LAMAX-S における組込み

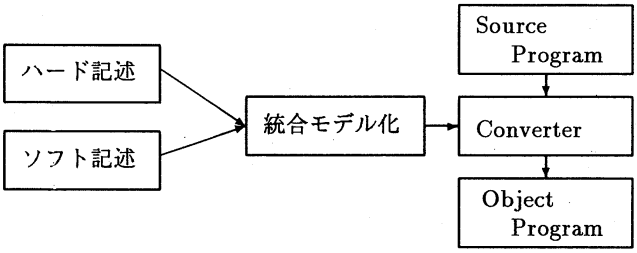
LAMAX-S は、FORTRAN に行列演算用の機能を取り入れた言語であり、プリプロセッサとして実現されている。1991 年秋に、パソコンで動作するプロトタイプ版が完成し、これを用いて記述性や実用性の評価が行われている。このプロトタイプ版の生成する行列演算用の FORTRAN コードは、ごく一般的な形式になっている。

我々は、本システムで作成されたチューニングルールを LAMAX-S の機構に取り入れ、このチューニングルールに従って命令を生成する予定である。この方式の導入により、LAMAX-S の目標の一つである、「パソコンからスーパコンピュータまで実行効率を保つ」ということが実現可能となる。



本文で提案したシステムは、非常に単純であるが、そこから得られる知識と利点は有益である。また、システムは非常に単純であるため、新たなチューニング手法に対する処理が容易である。たとえば、並列計算機で、行列の加算を偶数番地と奇数番地を分けて演算するコーディング方法が有効かどうかを試すことは容易である。

現実的には難しいであろうが、次のようにハードウェアおよびソフトウェアのスペック記述からそのコンピュータモデルを組立て、チューニング効果を推定するシステムを構築できれば申し分ないと考えている。



今後の課題として、次の項目が挙げられる。

- 本システムのより一層の自動化
- 本システムの結果の LAMAX-S への組入み
- さまざまな種類のコンピュータへの適応
- 実際の問題へ適用とその評価

参考文献

1. 内田智史、八巻直一、本郷茂、井上美明、唐澤豊：行列の数学的特性を重視した言語 LAMAX-S の設計思想とその処理系について、日本応用数理学会論文誌 Vol.2, No.3 1992
2. 寒川光：数値計算プログラミングにおけるデータ移動制御のためのブロック化アルゴリズム、情報処理学会論文誌 Vol. 33, No. 10 1992